

Conduite et gestion de projets informatiques : une introduction

G. Picard

SMA/G2I/ENS Mines Saint-Etienne

gauthier.picard@emse.fr

Septembre 2009



Plan

- Introduction
- Modèles et activités de développement
- Avant-Projet
- Suivi du projet
- Clôture du projet
- Activités transverses



Introduction

- A physician, a civil engineer and a computer scientist were arguing about what was the oldest profession in the world.
  - The physician remarked, « *Well, in the Bible, it says God created Eve from a rib taken out from Adam. This clearly required surgery, and so I can rightly claim that mine is the oldest profession in the world*. »
  - The civil engineer interrupted, and said, « *But even earlier in the book of Genesis, it states that God created the order of the heavens and the earth from out chaos. This was the first and certainly the most spectacular application of civil engineering. Therefore, fair doctor, you are wrong: Mine is the oldest profession in the world*. »
  - The computer scientist leaned back in her chair, and then said confidently, « *Ah, but who do you think created the chaos*? »

Logiciel

- Objet immatériel pendant son développement, très facile à modifier,
- Ses caractéristiques attendues sont difficiles à figer au départ et souvent remises en cause en cours de développement,
- Les défaillances et erreurs ne proviennent ni de défauts dans les matériaux ni de phénomènes d'usure dont on connaît les lois mais d'erreurs humaines, inhérentes à l'activité de développement,
- Le logiciel ne s'use pas, il devient obsolète (par rapport aux concurrents, par rapport au contexte technique, par rapport aux autres logiciels, ...),
- Le développement par assemblage de composants, des services, d'applications n'est pas encore généralisé dans le domaine logiciel (beans, EJB, composants, ... Web services, … EAI, …).

Génie logiciel

- Ingénierie du logiciel ⇔ Software Engineering
- Ensemble de théories, de méthodes, de techniques et d'outils pour la production et la maintenance de systèmes logiciels de qualité
- Domaine des 'sciences de l'ingénieur' dont la finalité est la conception, la fabrication et la maintenance de systèmes logiciels complexes, sûrs et de qualité ('Software Engineering')
- Art de la fabrication collective d'un système complexe, concrétisée par un ensemble de documents de conception, de programmes et de jeux de tests avec souvent de multiples versions.



Motivations

- Répondre à la 'crise du logiciel apparue dans les années 70 (prise de conscience que le coût du logiciel dépassait le coût du matériel)
- Répondre à la croissance de la taille et de la complexité des systèmes
  - besoins et fonctionnalités augmentent, évoluent
  - technologies en perpétuelle évolution
  - diversification des architectures
- Faire face aux délais de plus en plus courts,
- Gérer des équipes de plus en plus grosses, avec des compétences multiples



Préoccupations

- L'industrialisation de la production du logiciel :
  - organisation des procédés de production (cycle de vie, méthodes, notations, outils), organisation des équipes de développement, établissement de plan qualité rigoureux, etc.
- Des principes :
  - Rigueur et formalisation, Séparation des problèmes, Modularité, Abstraction, Anticipation des changements, Généricité, Construction incrémentale
- Règle du CQFD (**Coût Qualité Fonctionnalités Délai**)
  - Le système qui est fabriqué répond aux *besoins* des utilisateurs (correction fonctionnelle).
  - La *qualité* correspond au contrat de service initial.
  - Les *coûts* restent dans les limites prévues au départ.
  - Les *délais* restent dans les limites prévues au départ.

Qualité du logiciel : facteurs externes

- Correction (validité) : aptitude à répondre aux besoins et à remplir les fonctions définies dans le cahier des charges
- Robustesse (fiabilité) : aptitude à fonctionner dans des conditions non prévues au cahier des charges, éventuellement anormales
- Extensibilité : facilité avec laquelle de nouvelles fonctionnalités peuvent être ajoutées à un logiciel
- Compatibilité : facilité avec laquelle un logiciel peut être combiné avec d 'autres
- Efficacité : utilisation optimale des ressources matérielles (processeur, mémoires, réseau, …)
- Convivialité : facilité d 'apprentissage et d 'utilisation, facilité de préparation des données, facilité de correction des erreurs d 'utilisation, facilité d 'interprétation des résultats
- Intégrité (sécurité) : aptitude d 'un logiciel à protéger son code contre des accès non autorisés.

Qualité du logiciel : facteurs internes

- Ré-utilisabilité : Aptitude d ’un logiciel à être réutilisé, en tout ou en partie, pour d ’autres applications
- Vérifiabilité : aptitude d ’un logiciel à être testé (optimisation de la préparation et de la vérification des jeux d ’essai)
- Portabilité : aptitude d ’un logiciel à être transféré dans des environnements logiciels et matériels différents
- Lisibilité,
- Modularité.

Etat des lieux

Fonctionnent à la livraison 2%
Fonctionnent après modifications mineures 3%
Utilisés après modifications majeures 20%
Payés mais jamais livrés 30%
Jamais utilisés 45%

- IBM, 1992, 24 projets d'envergure en TI :
  - 55% de dépassement de budget
  - 68% de dépassement d'échéancier
  - 88% de ré ingénierie substantielle
- Standish Group, 1994, 8000 projets en TI :
  - Seulement 10% des projets sont livrés selon le budget et l'échéancier initial

http://www.standishgroup.com/sample_research/chaos_1994_1.php

Mythes du logiciel

- Mythes du client ou usager
- Mythes du développeur
- Mythes des gestionnaires

Les mythes du logiciel
Mythes de l’usager

Mythe	***Réalité***
Un énoncé général des objectifs est suffisant pour commencer. On verra les détails plus tard.	Une définition insuffisante des besoins des utilisateurs est la cause majeure d’un logiciel de mauvaise qualité et en retard
Les besoins du projet changent continuellement, mais ces changements peuvent être facilement incorporés parce que le logiciel est flexible	Les coûts d’un changement pour corriger une erreur augmentent dramatiquement dans les dernières phases de la vie d’un logiciel

Les mythes du logiciel
Mythes du développeur

Mythe	***Réalité***
• Une fois que le programme est écrit, et marche, le travail du développeur est terminé	• 50%-70% de l'effort consacré à un programme se produit après sa livraison à l'usager
• Tant qu'un programme ne fonctionne pas, il n'y a aucun moyen d'en mesurer la qualité	• Les revues de logiciel peuvent être plus efficaces pour détecter les erreurs que les jeux d'essais pour certaines classes d'erreurs
• Pour le succès d'un projet, le bien livrable le plus important est un programme fonctionnel	• ....

Les mythes du logiciel
Mythes du gestionnaire

Mythe	***Réalité***
• L'entreprise possède des normes, le logiciel développé devrait être satisfaisant	• Une configuration de logiciel inclue de la documentation, des fichiers de régénération, des données d'entrée pour des tests, et les résultats des tests sur ces données
• Les ordinateurs et les outils logiciels que l'entreprise possède sont suffisants	• ....
• Si le projet prend du retard, on ajoutera des programmeurs	

Maîtriser le développement

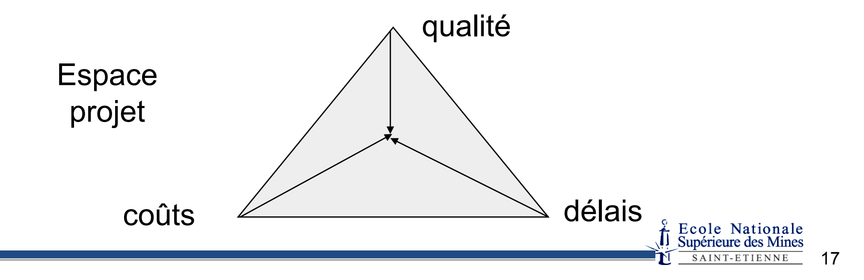
- Utiliser des techniques d'industrialisation (cf. calculettes, micros)
- Concevoir chaque logiciel comme une brique d'un projet (= travailler en mode projet)
- Les aspects d'évaluation des coûts et métrologie sont fondamentaux (CMM, ISO, SPICE,...)

Conduire le développement

- S'imposer des processus formels de développement
- processus d'assurance qualité
- des points de contrôle (milestones)
- méthode structurée, «phasée»
- des produits finis en fin de phase: inspection et validation après chaque phase du développement
- automatisé
- adaptable
- processus formel et exhaustif de tests
- technologie à jour (objets, Java, AGL,...)
- …

Projet informatique

- Ensemble d’actions mises en œuvre, afin de produire les résultats et fournitures définies en réponse aux objectifs clairement définis
- dans des délais fixés (date début et date de fin)
- mobilisant des ressources humaines et matérielles
- possédant un coût prévisionnel et des gains espérés



Acteurs d’un projet (1)

- **Maîtrise d’ouvrage :** personne physique ou morale propriétaire de l’ouvrage. Il détermine les objectifs, le budget et les délais de réalisation.
- **Maîtrise d’œuvre :** personne physique ou morale qui reçoit mission de la maîtrise d’ouvrage pour assurer la conception et la réalisation de l’ouvrage.

Acteurs d’un projet (2)

- Le Commanditaire
- Le Client
- Le comité directeur (moyen et gros projet)

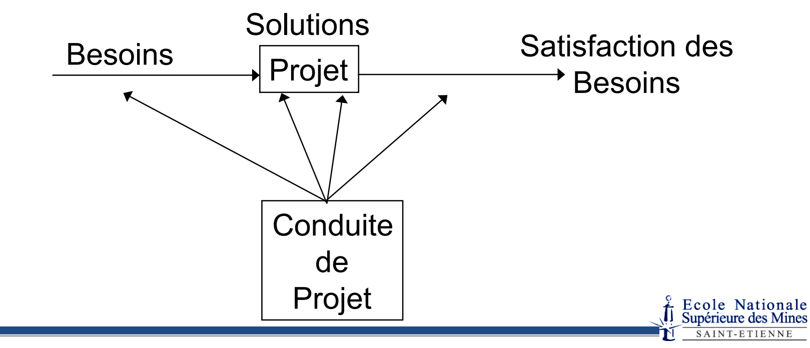
} Maîtrise d’ouvrage

- Le chef de projet
- L’équipe projet
- Les experts
- Le planificateur
- L’organisateur
- Le contrôleur
- L’innovateur
- L’investigateur
- Les utilisateurs

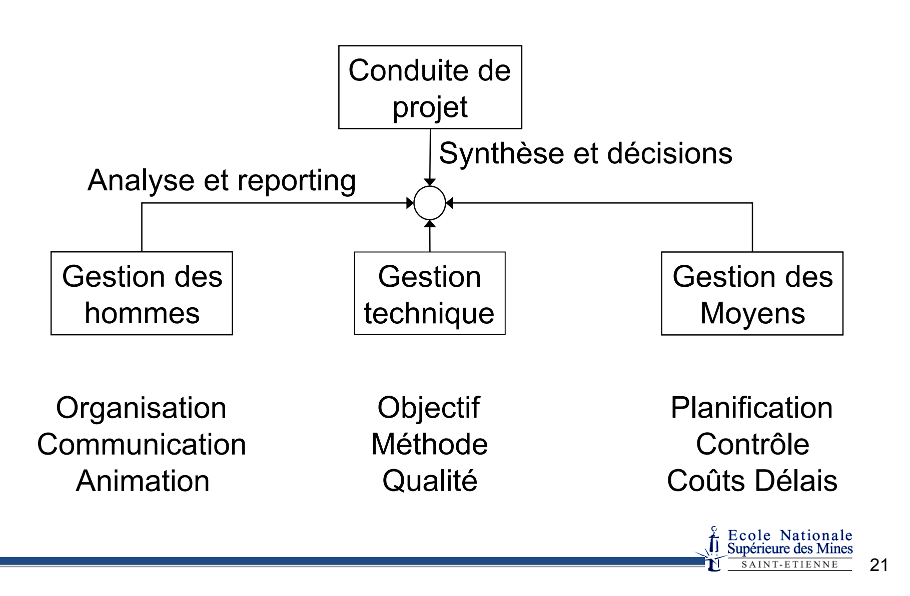
} Maîtrise d’œuvre

Conduite de projet (1)

Organisation méthodologique mise en œuvre pour faire en sorte que l’ouvrage réalisé par le maître d’œuvre réponde aux attentes du maître d’ouvrage dans les contraintes de délai, coût et qualité.



Conduite de projet (2)



Conduite et gestion de projet

- Processus difficile à maîtriser
  - ➟ Facteurs de risque :
    - coûts et les délais à respecter
    - technologies à maîtriser
    - ressources humaines à gérer
  - ➟ Pour réduire ces risques :
    - Définir des principes de base, communs à l'ensemble des projets afin de clarifier la terminologie
    - Coordonner les intervenants
    - Veiller à la cohérence des différentes activités

Conduite et gestion de projet

- La conduite de projet se situe à 2 niveaux
  - lors de la **conception** : fixer les objectifs, la stratégie, les moyens, l'organisation et le programme d'action
  - lors de la **réalisation** : s'assurer du bon déroulement du projet, de la qualité, du respect des délais et des budgets, faciliter les travaux de mise en œuvre et de maintenance

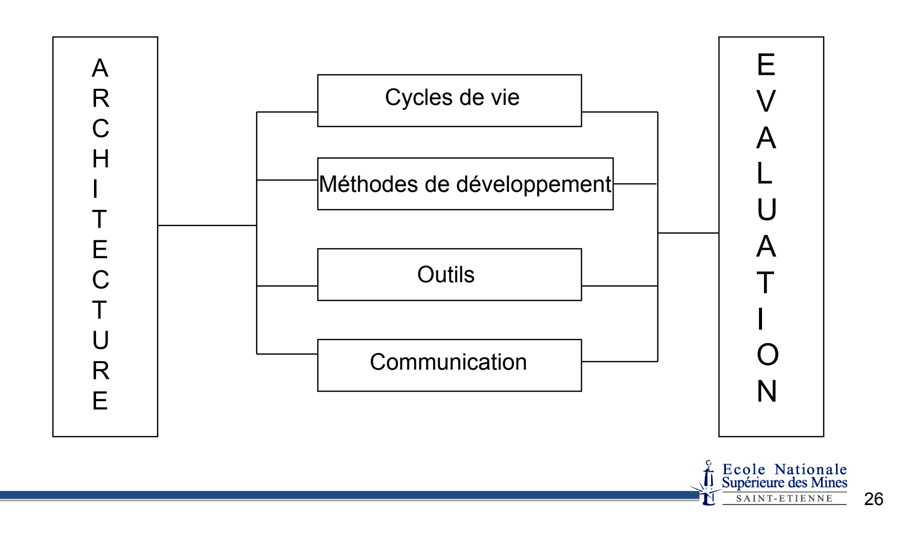
Conduite et gestion de projet *Modèles*

1) basés sur les livrables : modèles linéaires
   - Le processus de développement est divisé en étapes indépendantes, consécutives ou non
   - Chaque étape donne lieu à une revue et produit un document
2) basés sur le risque : modèle en spirale
   - Le modèle en spirale de Boehm met en œuvre une évaluation régulière des risques liés au projet permettant la mise en œuvre de solutions techniques pour annihiler ou contrer ces risques. Cette évaluation englobe les autres approches :
   - ➟ Un cycle de spirale utilise :
     - un modèle de développement en cascade (quand un risque d'intégration est identifié)
     - le prototypage quand le risque est lié à l'acceptation de l'interface utilisateur par le client, par exemple

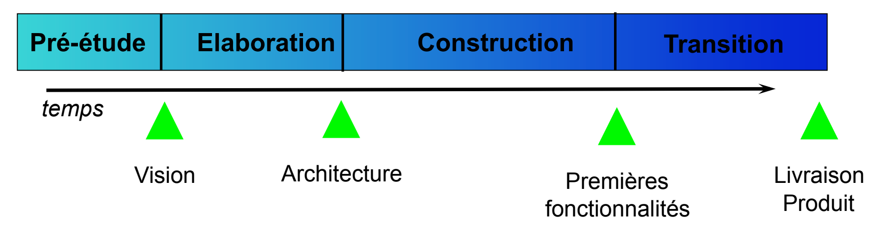
Conduite et Gestion de Projet
Référentiels

- La fabrication d'un logiciel de qualité respectant les contraintes de budget et de délais nécessite :
  - le choix d'une architecture
  - la mise en œuvre de méthodes, de techniques, de standards, des normes et des outils en vigueur au sein de l'organisation
- Ces méthodes, techniques, standards, normes, outils concernent aussi bien :
  - la production de composants logiciels (définition des besoins, conception, réalisation, tests,...)
  - le contrôle (planification, évaluation,...) du processus de production

Conduite et gestion de projet
Référentiels



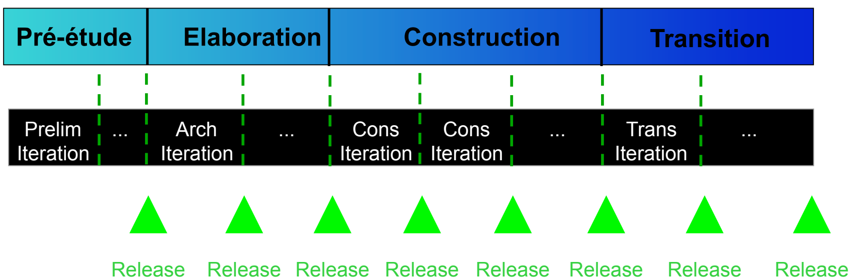
Cycle de développement
Phases



- Pré-étude : Définition de la portée du projet et développement des cas
- Vision : Glossaire, Détermination des parties prenantes et des utilisateurs, Détermination de leurs besoins, Besoins fonctionnels et non fonctionnels, Contraintes de conception
- Elaboration : Planification du projet, spécification des caractéristiques, des fondements de l'architecture
- Architecture : Document d'architecture Logicielle, Différentes vues selon la partie prenante, Une architecture candidate, Comportement et conception des composants du système
- Construction : Construction du produit
- Transition : Préparation du produit pour les utilisateurs

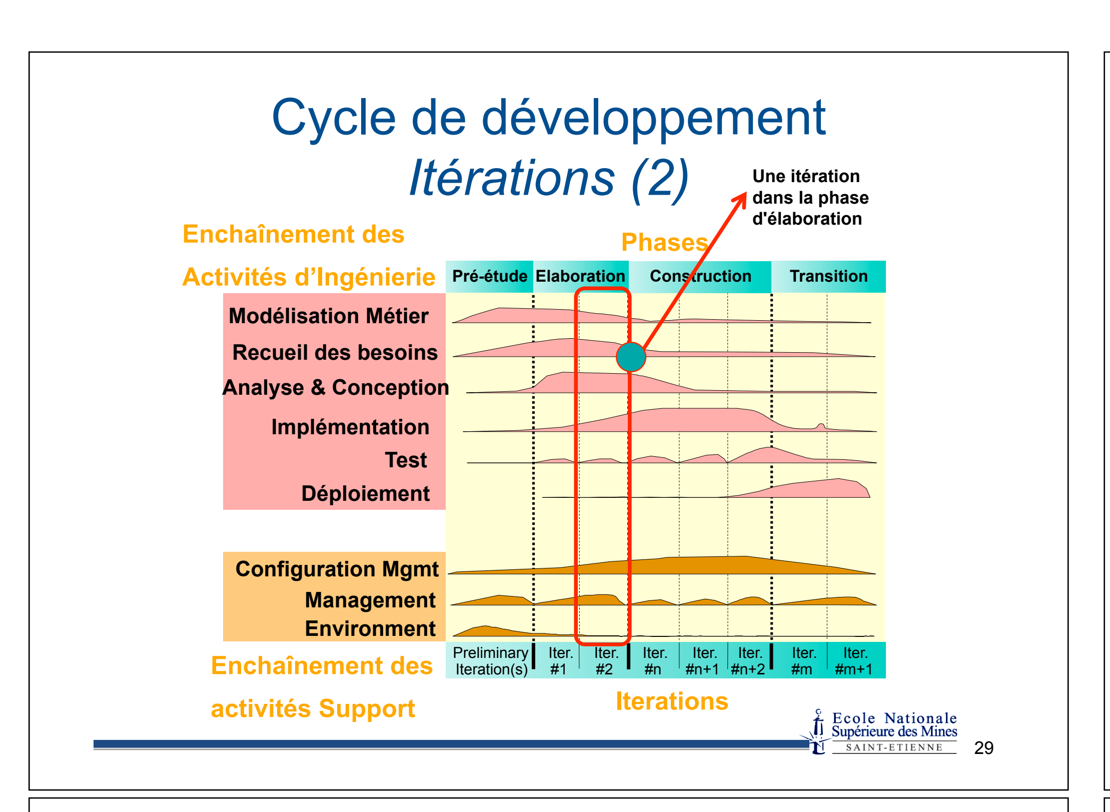


Cycle de développement
Itérations (1)



Une itération est une séquence d'activités selon un plan pré-établi et des critères d'évaluation, résultant en un produit exécutable

Cycle de développement
Itérations (2)



Cycle de développement
Intervenants

Gestionnaire du Projet

Montage du projet — Gestion du projet — Clôture du projet

Pré-étude	Elaboration	Construction	Transition

temps

Vision — Architecture — Premières fonctionnalités — Livraison Produit

Spécialistes techniques

Gestion de projet
Mise en œuvre

ORGANISER

PLANIFIER

- QUOI ?
- COMMENT ?
- QUI ? QUAND? COMBIEN?

Replanifier si nécessaire

MESURER

Ecarts

CONTROLER

Référentiel

EXECUTER

Réalisations

Prendre des actions correctrices

Conduite et gestion de projet
Causes de difficultés

- Qualité du produit, Estimation des risques, Mesures, Estimation du coût, Échéancier, …
- … Relation avec le client, Encadrement, …
- … Autres ressources, Contrôle du projet, …
- Communication
  - Fred Brooks remarque dans son livre «The mythical man-month» que s'il y a n employés sur un projet: on a n(n-1)/2 besoins de communication
- Humains
  - + un projet est vaste et complexe , + la conduite de projet s'éloigne du domaine de la technique pour se rapprocher de celui des relations humaines

Plan

- Introduction
- Modèles et activités de développement
- Avant-Projet
- Suivi du projet
- Clôture du projet
- Activités transverses

Modèles de développement

- Organiser les différentes phases du cycle de vie pour l'obtention d'un logiciel fiable, adaptable et efficace
- Guider le développeur dans ses activités techniques
- Fournir des moyens pour gérer le développement et la maintenance (ressources, délais, avancement, etc.)
- Plusieurs modèles sont proposés :
  - Modèle "code-and-fix"
  - Modèle (linéaire) en cascade
  - Modèle en V
  - Modèle en spirale
  - ...
  - Processus unifié

Modèle en cascade

- Atteinte de l'objectif par atteinte ordonnée de sous – objectifs. Les activités sont représentées dans des processus séparés.
- Processus séquentiel: Chaque étape doit être terminée avant que la suivante commence.
- Livrables:
  - À la fin de chaque étape, le livrable est vérifié et validé.
  - Vérification: le livrable est-il correct ?
  - Validation: est-ce le bon produit ? (Comparé à l'énoncé de l'étape).

Modèle en cascade

Modèle en V

- Amélioration du modèle en cascade
- Met en évidence la symétrie et la relation qu'il y a entre les phases du début du cycle de vie et celles de fin.
- Les phases du début doivent être accompagnées d'une planification des phases de fin
- Lors de la planification, on développe et documente les plans de test.

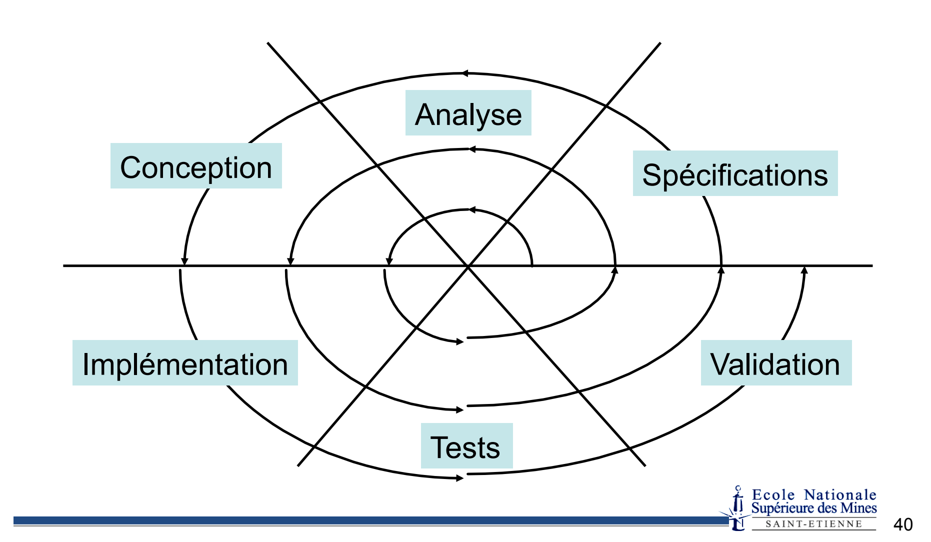
Modèle en V

Expression des besoins
Validation des besoins
Analyse et spécification
Validation fonctionnelle
Conception du système
Tests du système
Conception des composants
Tests des composants
Implémentation

Modèle en spirale

- Mise de l'accent sur l'évaluation des risques.
- A chaque étape, après avoir défini les objectifs et les alternatives, celles-ci sont évaluées par différentes techniques (prototypage, simulation, ...), l'étape est réalisée et la suite est planifiée.
- Le nombre de cycles est variable selon que le développement est classique ou incrémental.

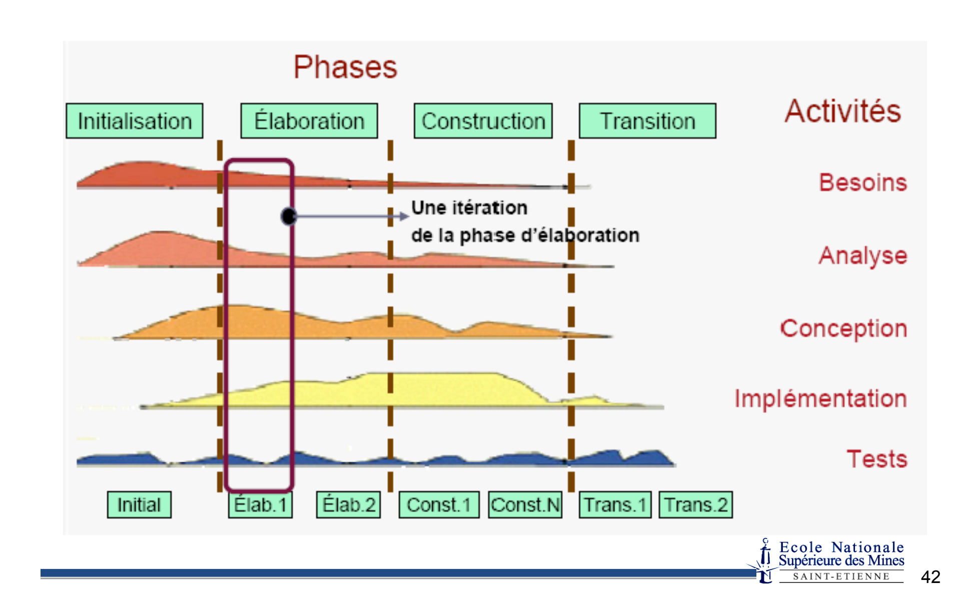
Modèle en spirale



Processus unifié

- Regroupement des activités à mener pour le développement d'un système logiciel, basé sur la notion d'objets.
- Piloté par les cas d'utilisation (bien comprendre les désirs et les besoins de ses futurs utilisateurs)
  - Un cas d'utilisation est une fonctionnalité du système produisant un résultat satisfaisant pour l'utilisateur. Les cas d'utilisation saisissent les besoins fonctionnels et leur ensemble forme le modèle des cas d'utilisation qui décrit les fonctionnalités complètes du système.
- Centré sur l'architecture (les différentes vues du système qui doit être construit)
- Itératif et incrémental
  - Itératif : croissance et l'affinement successifs d'un système par le biais d'itérations multiples, retours en arrière et adaptation cycliques
  - Incrémental : découpage du travail en plusieurs parties qui sont autant de mini-projets. Chaque mini-projet représente une itération ou étape de courte durée (1 mois) qui donne lieu à un incrément. Le résultat de chaque itération est un système testé, intégré et exécutable.

Processus unifié



Activités de développement

- Elles sont décrites de façon indépendante en indiquant leur rôle, utilisent et produisent des "artefacts"
- Selon le modèle, une activité peut jouer un rôle plus ou moins important et parfois ne pas exister du tout.
- Elles concernent :
  - Planification (Étude de la faisabilité)
  - Spécification des besoins (Requirement analysis)
  - Analyse (Spécification formelle)
  - Conception (Spécification technique)
  - Implémentation (Codage) et tests unitaires
  - Intégration et tests d'ensemble
  - Livraison
  - Maintenance

Planification

- Objectifs :
  - identification de plusieurs solutions et évaluation des coûts et bénéfices de chacune d'elles
- Activités :
  - simulation de différents scénarios de développement
- Résultats :
  - Rapport d'analyse préliminaire et un schéma directeur contenant :
    - la définition du problème et les différentes solutions étudiées, avec, pour chacune d'elles, les bénéfices attendus, les ressources requises (délais, livraison, etc.)

Spécification des besoins

- Objectifs :
  - définition de ce que doit faire le logiciel
- Activités :
  - Description du problème à traiter afin d'identifier les besoins de l'utilisateur, de spécifier ce que doit faire le logiciel : informations manipulées, services rendus, interfaces, contraintes
  - Mise en œuvre des principes : abstraction, séparation des problèmes, séparation des besoins fonctionnels
- Résultats : cahier des charges et plan qualité
  - un dossier d'analyse comprenant les spécifications fonctionnelles et non fonctionnelles du produit
  - une ébauche du manuel utilisateur pour les non informaticiens
  - une première version du glossaire contenant les termes propres au projet.
  - un plan de test du futur système (cahier de validation)



Analyse

- Objectifs :
  - Analyse détaillées de toutes les fonctions et autres caractéristiques que le logiciel devra réaliser pour l'usager, telles que vues par l'usager.
- Activités :
  - Répondre au « Que fait le système ? », Modélisation du domaine d'application
  - Analyse de l 'existant et des contraintes de réalisation
  - Abstraction et séparation des problèmes, séparation en unités cohérentes
- Résultats : Dossier d'analyse et plan de validation
  - Modèle du domaine
  - Modèle de l'existant (éventuellement)
  - Définition du modèle conceptuel.
  - Plan de validation, dossier de tests d 'intégration

Conception

- Objectifs :
  - Définition de l'architecture générale du logiciel. Spécification de la manière dont chacun des composants du logiciel sera réalisé et comment ils interagiront.
- Activités :
  - Répondre au « Comment réaliser le système »
  - Décomposition modulaire, définition de chaque constituant du logiciel : informations traitées, traitements effectués, résultats fournis, contraintes à respecter
- Résultats : dossier de conception + plan de test global et par module
  - proposition de solution au problème spécifié dans l'analyse
  - organisation de l'application en modules et interface des modules (architecture du logiciel),
  - description détaillée des modules avec les algorithmes essentiels (modèle logique)
  - structuration des données.

Implémentation

- Objectifs :
  - Réalisation des programmes dans un (des) langage(s) de programmation
  - Tests selon les plans définis lors de la conception
- Activités :
  - traduction dans un langage de programmation,
  - Mise au point (déboguage)
- Résultats : dossiers de programmation et codes sources.
  - Collection de modules implémentés, non testés
  - Documentation de programmation qui explique le code

Tests unitaires

- Objectifs :
  - test séparé de chacun des composants du logiciel en vue de leur intégration
- Activités :
  - réalisation des tests prévus pour chaque module
  - les tests sont à faire par un membre de l'équipe n'ayant pas participé à la fabrication du module
- Résultats :
  - résultats des tests avec les jeux d'essais par module selon le plan de test.



Intégration et test du système

- Objectifs :
  - Intégration des modules et test de tout le système
- Activités :
  - Assemblage de composants testés séparément
  - Démarche d'intégration (ascendante, descendante ou les deux)
  - Conception des données de tests
  - Tests Alpha : l'application est mise dans des conditions réelles d'utilisation, au sein de l'équipe de développement (simulation de l'utilisateur final)
  - Documentation des éléments logiciels
- Résultats :
  - Rapports de test
  - Manuel d'utilisation

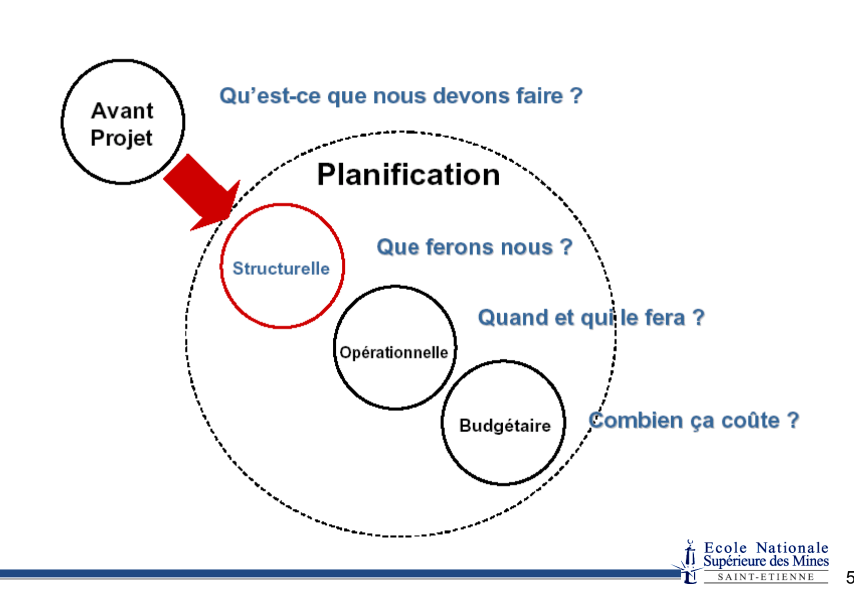
Livraison, maintenance, évolution

- Objectifs :
  - Livraison du produit final à l'utilisateur,
  - Suivi, modifications, améliorations après livraison.
- Activités :
  - Tests Bêta : distribution du produit sur un groupe de clients avant la version officielle,
  - Livraison à tous les clients,
  - Maintenance : corrective, adaptative, perfective.
- Résultats : la version finale du manuel utilisateur, les traces d'évolution du système, les rapport d'exploitation
  - Produit et sa documentation
  - Trace d'exploitation et d'évolution

Plan

- Introduction
- Modèles et activités de développement
- Avant-Projet
  - Estimation
  - Planification
- Suivi du projet
- Clôture du projet
- Activités transverses

Planification



Planification

- Outil incontournable pour la gestion du projet
- Il permet de :
  - définir les travaux à réaliser
  - fixer des objectifs
  - coordonner les actions
  - maîtriser les moyens
  - diminuer les risques
  - suivre les actions en cours
  - rendre compte de l'état d'avancement du projet

Planification structurelle

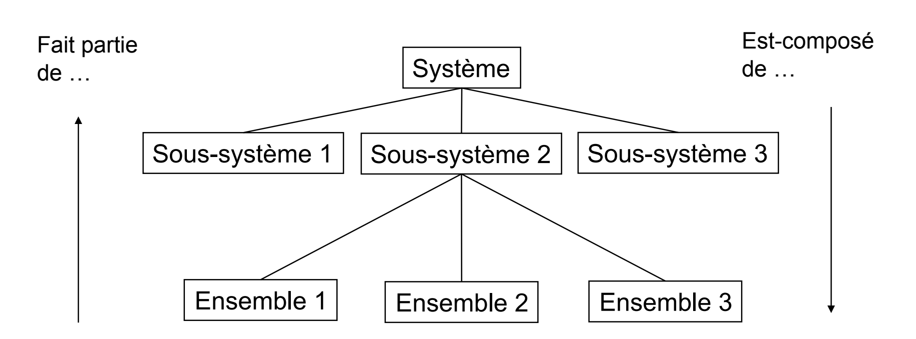
- Rôle :
  - Identifier les travaux à compléter
  - Traduire la définition du projet en une liste de tâches à accomplir
  - préparer une liste exhaustive, documentée et structurée des travaux dont l'accomplissement est nécessaire à la production des biens livrables du projet

➟ Constitution d'une base de données des travaux
  - Sert de base aux autres étapes de planification
  - Principal instrument de communication entre les intervenants
- Identification et description des lots de travail principaux
- Identification et description des tâches élémentaires

Planification structurelle *Etapes*

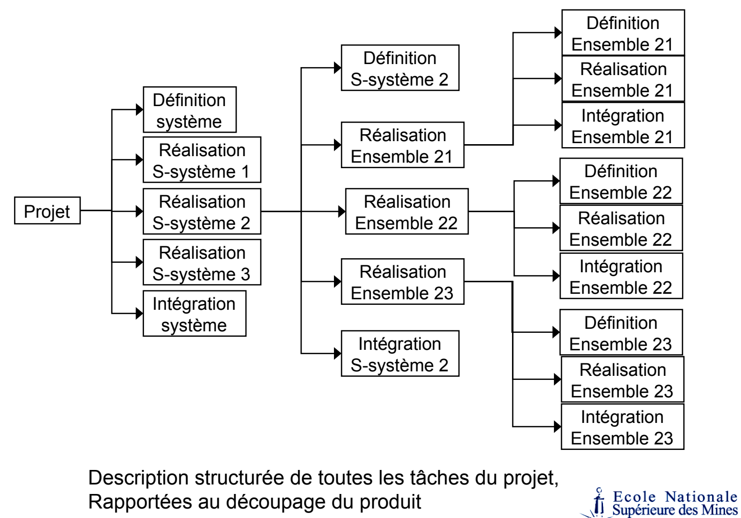
- Planification structurelle sommaire
  - Subdiviser le projet en lots de travail
  - Un lot = un bien livrable du projet
  - Toujours prévoir les lots de support pour tâches ponctuelles
- Planification structurelle détaillée
  - Subdiviser les lots de travail principaux
  - Jusqu'à l'identification de tâches élémentaires
  - Représentation à l'aide d'un organigramme de tâche (Work Breakdown Structure)
- Conformité et complétude
  - On doit avoir suffisamment confiance dans le caractère exhaustif de la liste des tâches pour être assuré que, une fois complétée de façon suffisante chacune des tâches élémentaires y apparaissant, le produit visé est effectivement réalisé et conforme aux exigences initiales

Planification structurelle
Product Breakdown Structure



Découpage du système en unités physiques hiérarchisées

Planification structurelle
Work Breakdown Structure



Description structurée de toutes les tâches du projet,
Rapportées au découpage du produit

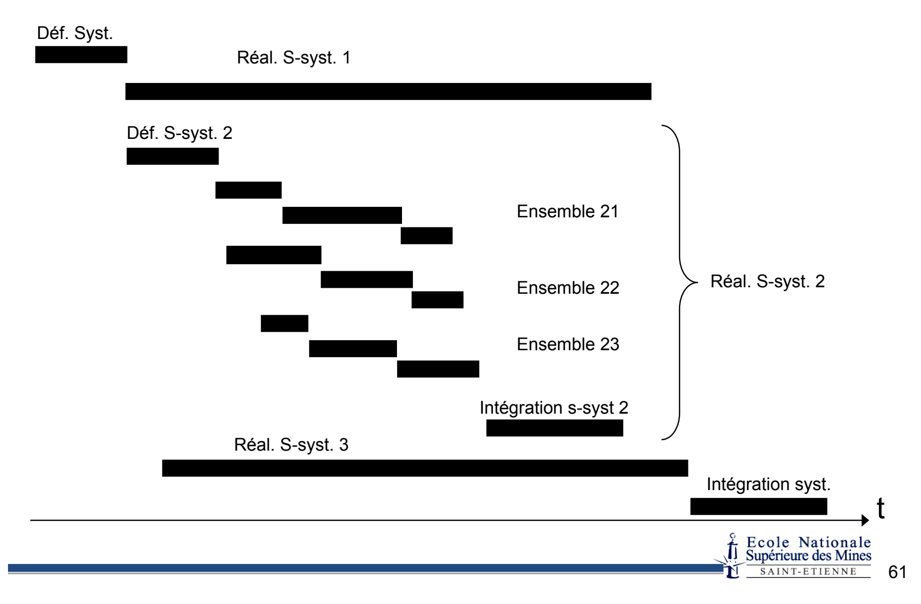
Planification opérationnelle

- Toute tâche est assignée à une personne
- Tout participant est informé de :
  - ses rôles et responsabilités
  - son degré d'autonomie et d'autorité
  - des rôles et responsabilités des autres
- Données de départ :
  - Organigramme technique
  - Processus de développement

Planification opérationnelle

- Rôle
  - Créer un réseau ordonnancé d'activités à partir des tâches de l'organigramme technique
  - Estimer de la durée d'une activité et des ressources requises pour la compléter
  - Identifier le chemin critique dans un réseau ordonnancé et calculer les marges totales, libres et d'indépendance
  - Utiliser les différents modes de présentation des résultats
- Caractéristiques
  - Forme la base pour la planification et la prédiction d'un projet
  - Facilite le choix des ressources pour compléter un projet à l'intérieur des échéanciers et du budget
  - Fournit les renseignements nécessaires pour prendre des décisions.
  - Identifie les dépendances entres les activités
  - Identifie le chemin le plus long : le chemin critique
  - Permet d'effectuer l'analyse des risques d'échéancier

Planification opérationnelle



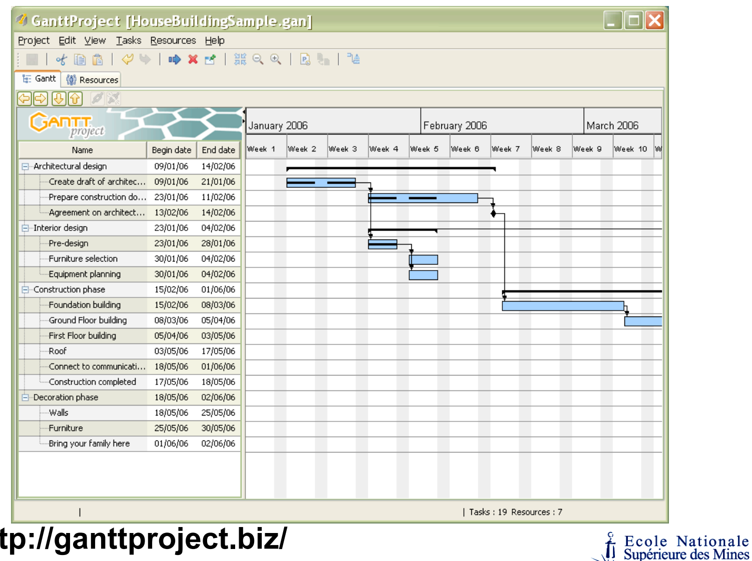
Planification opérationnelle

- Organisation dans le temps des activités
  - Activités/Dépendances :
    - Contraintes temporelles entre activités,
    - Structure logique des activités
  - Ressources associées aux activités
  - Durée d'une activité : durée dans le meilleur des cas, ajout d'un délai de garantie, pondération pour tenir compte de l'imprévu
- La planification est un processus dynamique tenant compte de la situation réelle, des nouvelles informations acquises

Planification opérationnelle

- Diagramme Pert
  - Graphe ordonné décrivant les contraintes de précédence logique des activités
    - Lister les tâches
    - Indiquer la charge de chacune
    - Préciser les liens de dépendance entre tâches
    - Classer les tâches selon leur rang
- Diagramme de Gantt
  - calendrier sur lequel chaque activité est représentée par une barre grisée débutant à la date de début au plus tôt et terminant à la date de fin au plus tard, sur laquelle glisse une barre blanche correspondant aux dates réelles de début et de fin

GanttProject



http://ganttproject.biz/

Estimations (1)

- Pourquoi ?
  - Connaître le coût d'une “vue de l'esprit” qui deviendra réalité … au bout d'un temps fini
- Quoi ?
  - L'effort de développement (coût), la durée du projet (temps), autre (équipement, voyage, formation), ajouter (la logique des calculs, les hypothèses)
- Quand ?
  - Tout au long du cycle de vie du projet
- Pièges à éviter
  - Faire trop précis (→ travailler avec des marges d'erreur importantes)
  - Sous-estimer (→ être exhaustif dans la liste des choses à estimer)
  - Sur-estimer (→ ne pas intégrer systématiquement tous les coûts possibles)
  - Confondre objectif et estimation (→ résister à “il ne faut pas que ça coûte plus de …”)
  - Vouloir tout estimer (→ savoir avouer son ignorance)



Estimations (2)

- Qualité de l'estimation
  - Rendue dans les délais, homogène en précision, honnête, complète, hypothèses explicites, réaliste, proche du coût réel
- Qualités de l'estimateur
  - Utile au client, organisé, objectif, compétent, créatif, réaliste, manie l'analogie
- Limites
  - manque de données historiques pour faire l'estimation, nouvelles technologies, manque d'expérience en estimation, oublis, productivité n'est pas « 40 heures/semaine », optimisme non fondé



Estimation
Démarche et conseils

- Démarche
  - Entrées : objectifs techniques, objectifs de délais, environnement, période, historique, références
  - Sorties : estimation
  - Itérations : augmenter l'information et comparer avec le résultat
- Conseils
  - Toute information est bonne à prendre et à classer
  - Les projets déjà réalisés sont la meilleure source (→ tableau de bord)
  - Exploiter les offres de ses fournisseurs
  - Adhérer aux associations professionnelles
  - Lire les revues spécialisées de sa profession
  - Être organisé, être créatif, affûter ses outils
  - Constituer une check-list
  - Vérifier ses estimations
  - Remettre à jour ses données

Estimation
Méthodes

- Par analogie
  - Exploration des expériences passées, catalogue des projets et estimations passées. Ce qui est analysé concerne : taille, durée, effort, complexité, coût
- Modèle paramétrique
  - Les estimations sont basées sur des modèles mathématiques reposant sur divers paramètres : COCOMO, SLIM, PRICE-S, SoftCost, …
- Oracle
  - Equipe d'experts, atteinte d'un consensus par négociation
- PERT
  - Estimations reposant sur l'hypothèse d'une répartition normale des estimations
  - Réaliser plusieurs estimations avec une méthode “par analogie” ou “oracle” → la pire (l), la moyenne (m), la meilleure (h)
  - Effort = (l+4m+h)/6
- Bottom-up
  - Les estimations par analogie, PERT, paramétrique, oracle sont faites par activité ou composant élémentaire
  - Puis consolidées jusqu'au sommet du projet
- Aucune technique n'est meilleure ou pire que les autres.
  - Utiliser plusieurs techniques en parallèle et comparer les résultats : si trop de différence, augmenter la quantité d'informations prises en compte.

Estimation
Taille du logiciel

- Point fonction mesure du montant de fonctionnalité en s'appuyant sur :
  - 5 type de fonctionnalités :
    - Input, Output, Inquiry, Internal Logical File, External Interface File
- FC = nombre de fonctions
- Ajuster selon leur complexité ci à partir de 14 facteurs notés de 0 (pas d'influence) à 5 (fondamental)
  - Communication par message, distribution de données ou de fonctions, haut taux de transaction, calcul complexe, conception multi-sites, conception facilement maintenable, …
- FP = FC * PCA PCA = 0,65+0,01 Somme(ci)
- KLSL = -5 + 0,2 FP

Estimation
Types de fonctionnalités

- Input (entrée utilisateur)
  - Entrée de donnée ou de contrôle qui requiert un traitement
  - Écrans, transactions, fichier de données, etc.
- Output (sortie utilisateur)
  - Sortie de donnée ou de contrôle après un traitement du système
  - Écrans, transactions, fichier de données, etc.
- Internal files (fichiers internes)
  - Regroupement logique de données ou de contrôle interne au système
  - Bases de données, répertoires, etc.
- External interface files (fichiers externes)
  - Fichier ou exécutable qui sortent des limites du système
  - Bibliothèques, bases de données externes, paquetages génériques, etc.
- Inquiry (requêtes)
  - Entrée ou sortie d'une requête demandant une réponse immédiate du système
  - Interruptions, appels, etc.

Estimation
Facteurs d'influence

- Interconnections
- Distribution des fonctions
- Performance
- Utilisation opérationnelle lourde
- Taux de transaction
- Entrée de données en ligne
- Facilité d'utilisation
- Mise à jour en ligne
- Traitements complexes
- Réutilisation du code
- Facilité d'installation
- Facilité d'opération
- Sites multiples
- Flexibilité

Effort ? Durée ? Taille ?

- **Effort** ou charge : quantité de travail nécessaire, indépendamment du nombre de personnes qui vont réaliser ce travail
  - Permet d'obtenir un coût prévisionnel
  - S'exprime en homme/jour, homme/mois ou homme/année
  - Un homme/mois (HM) représente l'équivalent du travail d'une personne pendant un mois, généralement 20 jours
  - Un homme/mois (HM)=152 heures de travail par mois
- Exemple: Un projet de 60 mois/homme représente l'équivalent du travail d'une personne pendant 60 mois. Si on évalue le coût du mois/homme à 50 K€ en moyenne, le projet sera estimé à 3 M€

Effort ? Durée ? Taille ?

- On mesure la taille des projets à leur charge
- Si Charge < 6 HM : très petit projet
- Si 6 HM< Charge <12 HM : petit projet
- Si 12 HM< Charge <30 HM : projet moyen
- Si 30 HM < Charge < 100 HM : grand projet
- Si Charge > 100 HM : très grand projet

Effort ? *Durée* ? Taille ?

La **durée** dépend du nombre de personnes

- 60 HM peut correspondre à
  - 1 personne pendant 5 ans
  - 5 personnes pendant un an
  - 10 personnes pendant 6 mois
  - 60 personnes pendant 1 mois

Effort ? Durée ? *Taille* ?

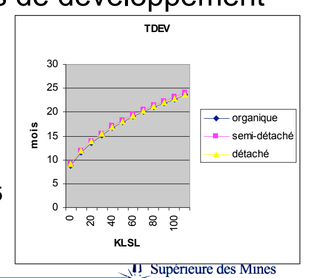
- **Homme-mois** = unité de mesure de l'effort
  - Un homme pendant un mois
  - Deux hommes-mois = 2 hommes pendant 1 mois, ou 1 homme pendant deux mois
- Selon Brooks : *« L'homme-mois comme unité pour mesurer la taille d'un travail est un mythe dangereux et trompeur. Il implique que les hommes et les mois sont interchangeables. Les hommes et les mois sont des biens interchangeables seulement lorsqu'une tâche peut être partitionnée entre plusieurs employés sans qu'il faille une communication entre eux »*.

Estimation *COCOMO*

- Modèle paramétrique http://sunset.usc.edu/research/cocomosuite/index.html
- Hypothèse : les besoins du logiciel sont relativement stables, le projet est géré à la fois par le client et par le fournisseur
- Formule d'estimation : Effort = $A\ (KLSL)^b$
  - KLSL : Kilo Lignes Sources Livrées : ligne source quelque soit le nombre d'instructions par ligne, sans tenir compte des commentaires ni du logiciel support
  - A et b estimées à partir de l'analyse des historiques
  - A et b dépendent des trois classes de projet
    - Organique : petites équipes (faible communication, distribution efficace du travail, …), environnement stable, applications bien comprises
    - Semi-détaché : équipe de taille moyenne (personnes expérimentées débutants), problèmes ne sont pas tous maîtrisés
    - Détaché : grande équipe, répartie, nouvel environnement

Estimation
COCOMO (simple)

- HM : Homme-mois = 152 h
  - Organique : HM = 2,4 $(KLSL)^{1,05}$
  - Semi-détaché : HM = 3,0 $(KLSL)^{1,12}$
  - Détaché : HM = 3,6 $(KLSL)^{1,20}$
  - Attention : nombre de personnes employées sur un projet n'est pas uniforme pendant le temps de développement
    - Effectif croît jusqu'à l'implémentation, décroît ensuite
- TDEV : temps de développement
  - Organique : TDEV = 2,5 $(HM)^{0,38}$
  - Semi-détaché : TDEV = 2,5 $(HM)^{0,35}$
  - Détaché : TDEV = 2,5 $(HM)^{0,32}$



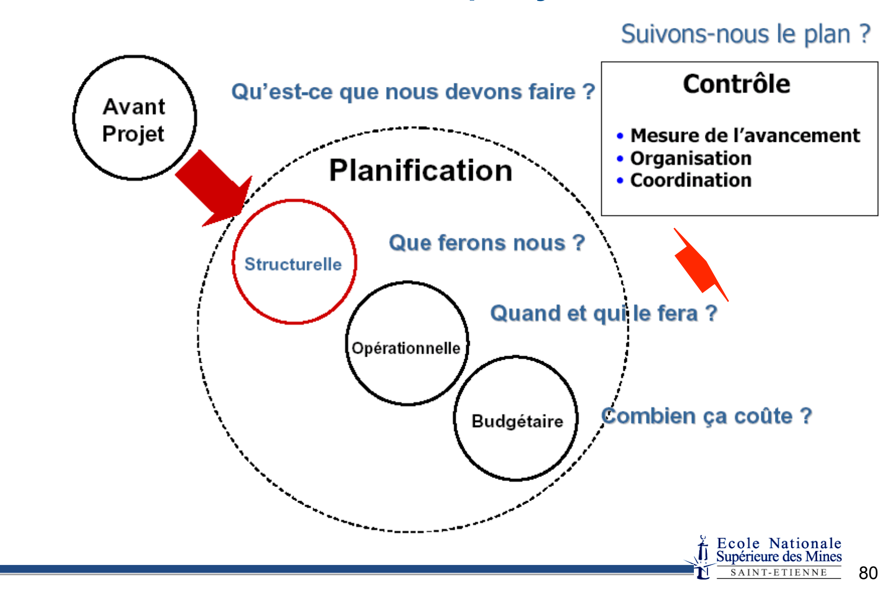
Estimation
COCOMO (intermédiaire)

- Point de départ : HM et TDEV du modèle simplifié
- Introduction de quinze facteurs correctifs, valués de VeryLow à XtraHigh
  - Pour le projet :
    - Fiabilité requise du logiciel
    - Taille de la base de données
    - Complexité du produit
  - Pour les contraintes de l'environnement
    - Contraintes de temps d'exécution
    - Contraintes de place mémoire
    - Stabilité de la machine virtuelle (matériel + logiciel) sur lequel le logiciel est développé
    - Système de développement interactif ou non
  - Pour le personnel
    - Aptitude à l'analyse
    - Expérience du domaine
    - Expérience de la machine virtuelle
    - Aptitude à la programmation
    - Expérience du langage
  - Pour les méthodes
    - Méthode de programmation moderne
    - Outils logiciels
    - Durée de développement

Plan

- Introduction
- Modèles et activités de développement
- Avant-Projet
- Suivi du projet
- Clôture du projet
- Activités transverses

Suivi de projet



Suivi de projet

- Mettre en place un processus de suivi et de revues régulières entre le chef de projet et les membres de l'équipe
- Un "journal de bord" est tenu à jour. Il permet de garder une trace :
  - des informations communiquées
  - des problèmes rencontrés
  - des décisions prises
  - des responsables désignés pour mener à bien les actions
  - la date de réalisation de l'action



Suivi de projet

- Cette fonction consiste à évaluer la situation réelle du projet, à la comparer à la situation prévue au plan d'exécution et à prendre les décisions nécessaires pour corriger la situation, si des écarts sont observés ou prévus
- La maîtrise des ressources et la gestion de la qualité du produit :
  - sont des fonctions en cours de réalisation du projet quelle que soit la phase atteinte dans la progression du projet
  - impliquent une base de comparaison que constitue le plan de réalisation, produit de la planification du projet et de l'utilisation des ressources



Maîtrise des ressources

- La maîtrise des ressources implique :
  - capacité d'expliquer les difficultés rencontrées au plan technique
  - capacité d'expliquer les retards et les dépassements de coût
  - capacité de proposer des mesures correctives, d'en évaluer les répercussions et de les mettre rapidement en œuvre
  - capacité à répondre à des conditions changeantes du milieu (le projet, son environnement)
- Cette capacité demande d'avoir des points de repère
  - C'est la planification du projet

Contrôle

- Activité d'acquisition des informations sur la progression du projet
  - Ce qui est complété
  - Les ressources effectivement utilisées
  - La date de début et de fin
- Ce qui en en cours : % d 'avancement
  - La date de début, ressources utilisées : matériaux, équipement, main d'œuvre
- Questions à résoudre:
  - Quoi documenter ? À quelle fréquence ?
  - Avec quelle résolution ? Problèmes rencontrés ?

Suivi de l’avancement *Analyse*

- But : vérifier si la situation actuelle est telle que prévue
- Compilation des informations recueillies
  - Calcul des coûts effectivement engagés et déboursés
  - Validation de l ’estimé du % d ’avancement
  - Nature exacte des problèmes rencontrés (recherche des causes
- Analyse prévisionnelle - valeur acquise
  - Comparaison avec la situation prévue
- Sélection de mesures correctives
  - Proposition et analyse de l ’effet de mesures correctives
  - Recommandations

Suivi de l’avancement *Causes d’écart (1)*

- Performance technique
  - Occurrence d ’un problème technique imprévu
  - Difficultés techniques majeures dont la mise en relation de diverses composantes
  - Problème de fiabilité dans les matériaux, les équipements achetés
  - Changement imposé par le client
  - Apparition d ’un nouveau produit sur le marché
  - Révision des spécifications techniques
- Coûts
  - Difficulté de financement
  - Difficultés techniques imposant l ’utilisation de plus de ressources humaines ou d ’équipement
  - Majoration des coûts des matériaux, de la main-d’œuvre, de l’énergie, etc.
  - Monitoring erroné
  - Délai dans la mise en œuvre des mesures correctives
  - Estimation initiale incorrecte

Suivi de l’avancement *Causes d’écart (2)*

- Échéanciers
  - Durée plus longue que prévue pour compléter une activité, pour résoudre un problème technique
  - Durée requise pour résoudre un problème nouveau
  - Mauvaise estimation de la durée des activités à réaliser
  - Pénurie de ressources humaines, matérielles et d’équipement
  - Répercussions des retards de réalisation des activités qui précèdent sur la durée des activités à venir, sur leur programmation, etc. (Boucle de rétroaction positive)
- Mise en œuvre
  - Approbation des mesures retenues
  - Communication aux personnes concernées
  - Mise en application

Suivi de l’avancement *Conseils élémentaires*

- Toujours donner l’heure juste
- S’assurer que le coût du contrôle, de l’analyse et de la mise en œuvre demeure inférieur aux bénéfices espérés du suive et du contrôle des ressources
- Ne prendre que les informations pertinentes à la maîtrise des ressources et de la qualité du produit
- Vérifier que le contrôle et l’analyse se font rapidement pour que les mesures correctives demeurent d’actualité
- Organiser le contrôle autour des biens livrables

Plan

- Introduction
- Modèles et activités de développement
- Avant-Projet
- Suivi du projet
- Clôture du projet
- Activités transverses



Clôture de projet (1)

- Inévitablement, les projets se terminent ; il est dans la définition même d'un projet qu'il ne dure qu'un temps précis dans la vie d'une organisation. Les façons dont les projets se terminent peuvent toutefois varier.
- Fin « normale » d'un projet
  - La plupart des projets se terminent favorablement avec la livraison du produit ou du système au client; ce client peut être à l'interne de l'organisation, projet d'implantation d'équipement dans une usine, ou à l'externe, projet de construction, projet de sous-traitance industrielle.
- Fin « normale » d'un projet et intégration à l'organisation
  - Dans certains cas de projets, surtout lorsque le client est interne, il arrive très fréquemment qu'on invite les membres de l'équipe à devenir ou redevenir membres à part entière à l'organisation. On parle donc d'intégration des résultats et des ressources du projet.
- Fin d'un projet avorté
  - Il peut arriver qu'on doive arrêter un projet pour des questions techniques, budgétaires ou légales. Des procédures doivent alors être prises pour compenser, s'il y a lieu, la ou les parties lésées.

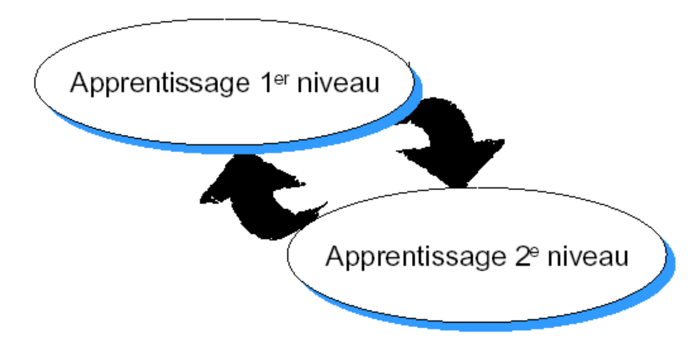


Clôture de projet (2)

- En situation normale, « clôturer » un projet désigne une série d'activités que doit réaliser les responsables du projet. L'utilisation de listes de vérification est fréquente lors de la fermeture de dossiers.
- S'assurer de la fin de l'ensemble des travaux, incluant les tâches en sous-traitance
- Validation du client comme quoi il a reçu le produit/système et les autres livrables
- S'assurer que la documentation est à jour et que les rapports de clôture ont été réalisés (si requis)
- Régler les dernières transactions financières (facturation)
- Relocalisation du personnel, des équipements, des matériaux
- Consolider la documentation à conserver

Ecole Nationale Supérieure des Mines SAINT-ETIENNE

Evaluation (1)



Ecole Nationale Supérieure des Mines SAINT-ETIENNE

Evaluation (2)

- Accroître la réactivité de l'équipe (identifier rapidement les problèmes et apporter des solutions)
- Agir sur les interactions performance coût échéanciers
- Améliorer la performance du projet
- Identifier des opportunités de développement technique
- Évaluer la qualité du projet / du produit
- Viser l'efficience
- Faire valoir rapidement les « bons coups »
- Maintenir et/ou améliorer la relation avec le client
- Confirmer l'engagement de l'organisation envers le projet



- Mieux comprendre le lien projet organisation
- Améliorer les processus de gestion de projet
- Améliorer les conditions dans lesquels se réalisent les projets
- Identifier les forces et faiblesses des intervenants, des systèmes
- Favoriser l'avancement des acteurs projets

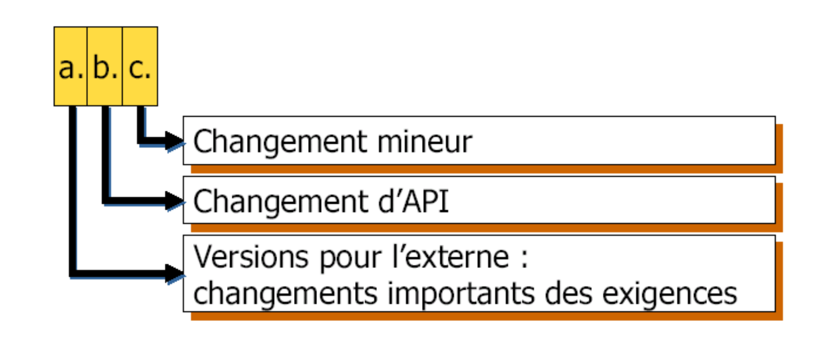
Plan

- Introduction
- Modèles et activités de développement
- Avant-Projet
- Suivi du projet
- Clôture du projet
- Activités transverses
  - Gestion de configurations
  - Documentations
  - Les outils
  - Les Hommes

Gestion des versions et des évolutions (1)

- Numérotation à trois chiffres :
  - 1er chiffre : Numéro de versions majeures du produit, don't la sortie s'accompagne de progrès importants au niveau des fonctionnalités, et/ou changement notable d'environnement d'utilisation ou de portabilité
  - 2ème chiffre : numéro des version mineures. L'incrément est réalisé à chaque fois que l'équipe de développement libère une version du produit qui corrige des bugs attendus par les clients (mais non bloquants), et apporte des modifications légères
  - 3ème chiffre : numéro des corrections, versions résultant de la maintenance
- Version Alpha : version terminée en cours de test et de revue de qualité
- Version Béta : version alpha validée en test auprès d'un panel de clients privilégiés

Gestion des versions et des évolutions (2)



Documentations

- Documentation de gestion du projet
  - Plannings, plans, estimations
  - Rapports
  - Définitions de standards
  - Documents de travail
  - Courriers (méls)
- Documentation Technique
  - Utilisateur : Manuel d'installation, manuel d'administration, manuel d'utilisation, manuel de référence
  - Système : cahier des charges, analyse et conception du système, architecture du système, archivage des programmes et des listings, documents de validation, documents de tests, guide de maintenance



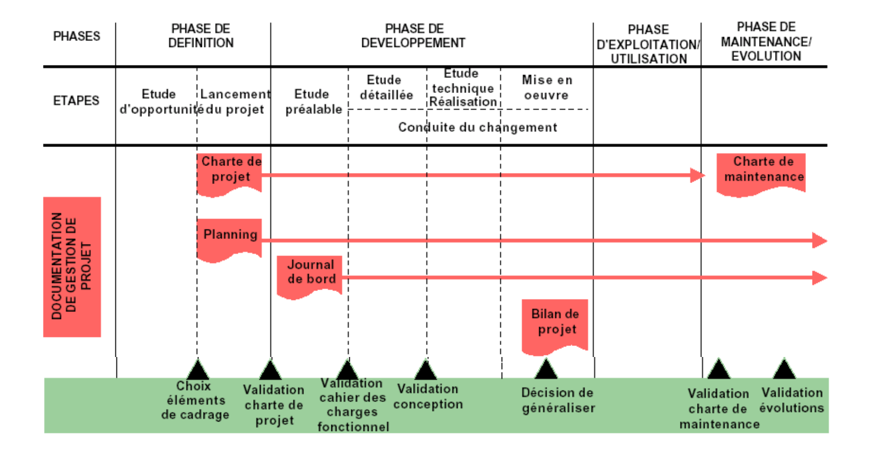
Documentations
Structure d'un document

- Un document comporte nécessaire une page d'en tête :
  - Qui le situe dans le projet (référence de projet, auteurs, catégorie du document)
  - Qui décrive sommairement son contenu (titre parlant, résumé)
  - Qui le date et donne sa version
  - Qui en permette l'archivage (mot clef, type de document)
  - Qui décrive les standards auquel le document est supposé se conformer
  - Qui en décrive le copyright
  - Qui en décrive la circulation souhaitée (nominatif et/ou classement de confidentialité)
  - Qui donne un contact pour questions ou remarques
  - Qui dise s'il s'agit d'une version préliminaire (de travail) ou définitive
  - …

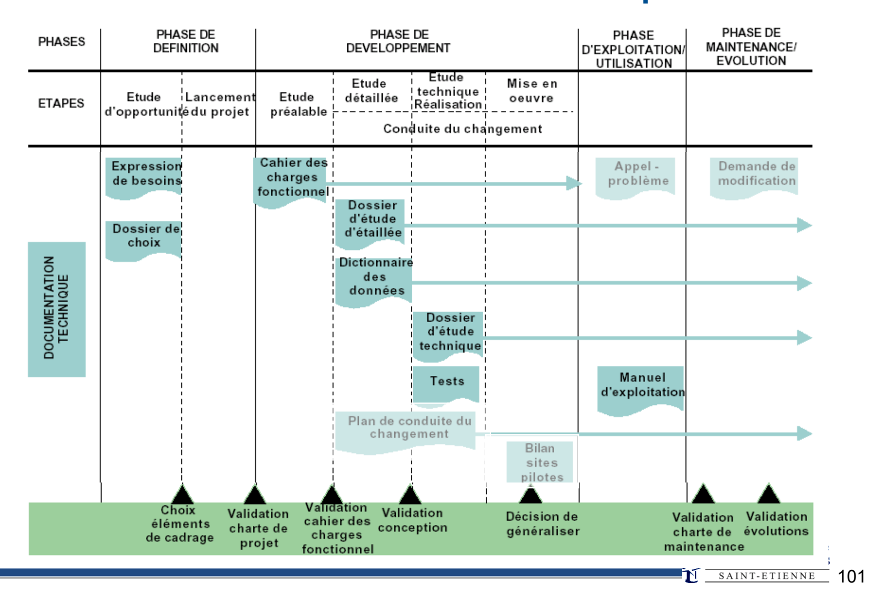
Documentations
Style rédactionnel

- Séparer clairement les paragraphes qui peuvent être perçus comme des réponses aux questions quoi ?, par qui ?, où ?
- Identifier des niveaux de texte correspondant à des lectures plus ou moins détaillées. Trois niveaux : titre, corps principal de texte, texte
- Mentionner en notes de bas de page les considérations à caractère anecdotique qui même si elles éclairent le sujet perturbent la compréhension d'une phrase
- Mettre en évidence la première apparition d'un terme dans le texte, et surtout une mention qui le définit, par exemple, en utilisant des caractères gras. Une définition ne doit pas pouvoir échapper à l'attention, même lors d'une lecture rapide
- Écrire des phrases et des paragraphes courts
- Ne pas utiliser de double négation
- Utiliser des formes verbales actives, impératives et le présent
- Avoir une bonne orthographe et une bonne grammaire
- Définir les termes utilisés : un glossaire doit impérativement accompagner tout document
- Se répéter si nécessaire
- Donner des références explicites.

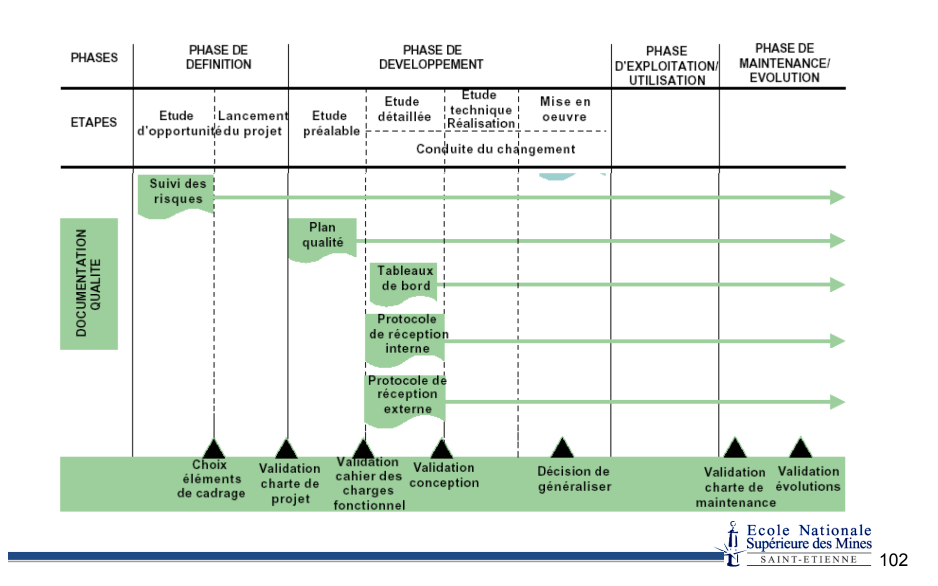
Documentations
de gestion de projet



Documentations techniques



Documentations qualité



Document d'analyse

1. Vision générale
2. Spécification préliminaire
3. Définition des cas d'utilisation
4. Spécification détaillée
5. Cas d'utilisation
6. Exemples
7. Collaborations
8. Diagrammes d'état
9. Graphes d'activité

Document d'analyse *Vision générale*

1.1 Positionnement

– This chapter describes the situation of the analysis document (positioning with regard to other analysis documents, requirements specifications, indication of associated software parts)

1.2 Objectifs

– This chapter describes the aim of this specification, fundamental needs met and the overall specification plan

1.3 Documents de référence

– This chapter provides the list of documents on which the current document is based, and to what extent

Document d'analyse : *Spécification préliminaire*

2.1 Dictionary

– This provides the list of terms used in the document, accompanied by their definition.

2.2 Overview of the application

– This chapter contains:
  - summary and description notes of the document package
  - class diagrams (with their description notes)

2.3 Summary

– This chapter contains the list of:
  - packages annotated {usecase} (with their summary notes)
  - packages not annotated {usecase} (with their summary notes)
  - classes (with their summary notes)
  - interfaces (with their summary notes)
  - referenced elements (with their summary notes)



Document d'analyse : *Définition des cas d'utilisation*

User packages

The "Definition of the use cases" chapter contains:

– packages annotated {usecase}.

Document d'analyse *Spécification détaillée*

4.1 Non-user packages

4.2 Classes

4.3 Interfaces

4.4 Referenced packages

4.5 Referenced classes

4.6 Referenced interfaces

Structure du document

1 Overview
1.1 Situation of this specification
1.2 Objectives of this specification
1.3 Reference documents
2 Preliminary specification
2.1 Dictionary
2.2 Overview of the application
2.3 Summary
3 Definition of the use cases
User packages
4 Detailed specification
4.1 Non-user packages
4.2 Classes
4.3 Interfaces
4.4 Referenced packages
4.5 Referenced classes
4.6 Referenced interfaces
5 Use cases
6 Examples
7 Collaborations
8...State machines
9...Activity graphs

Les outils

- Outils dédiés à des tâches spécifiques
- Ateliers de génie logiciel (AGL) :
  - Analyse et conception
  - Programmation, prototypage ou développement rapide (RAD)
  - Construction d'interface homme-machine
  - Vérification
  - Documentation, version, collaboratif
- Environnement intégrés pour un support à tout le développement



Les Hommes (1)

- Développement logiciel est une tâche humaine et créative
- Travail en groupe : Travail collaboratif, Productivité personnelle, Disponibilité d'outils de travail collaboratif
- Structure homogène (petits projets) :
  - structure de l'équipe reflète la structure du produit, chaque membre réalise une partie du projet
  - Bonne communication entre les membres, continuité du projet est facile à assurer
- Structure spécialisée (grands projets) :
  - Structuration selon les spécialités



Les Hommes (2)

- Rôle du chef de projet
  - Compétences techniques
    - Spécification
    - Architecture
    - Outils de développement
    - Tests
  - Compétences administratives et organisationnelles
    - Gestion administrative
    - Allocation de ressources
    - Animation des équipes
  - Trop pour une seule personne
    - Deux responsables (technique et administratif)

Les Hommes (3)

- Programmation impersonnelle
  - Pas de propriété personnelle (pas de lien affectif entre le module et la personne)
  - Propriété collective (présentation standardisée : mise en page, commentaires, …)
  - Tout programme contient des erreurs
  - En découvrant une erreur, on ne blâme pas une personne particulière, mais on rend un service à l'équipe
  - Plus tôt on découvre les erreurs, moins coûteuse est la correction

Rôles au sein d'un projet

- Programmers (system engineers)
  - Technical lead, architect, programmer, Sr. programmer
- Quality Assurance (QA) engineers (testers)
  - QA Manager, QA Lead, QA staff
- DBAs
  - DB Administrator, DB Programmer, DB Modeler
- CM engineers (build engineers)
- Network engineers, System Administrators
- Analysts (business analysts)
- UI Designers
- Information Architects
- Documentation writers (editors, documentation specialist)
- Project manager
- Other
  - Security specialist, consultants, trainer



Rôles au sein d'un projet

- Décider lesquels sont nécessaires pour votre projet
- En fonction de ce que vous êtes en train de développer :
  - How big is it?
  - Is it UI intensive? Data intensive?
  - Are you installing/managing hardware?
  - Do you need to run an operations center?
  - Is it in-house, contract, COTS, etc?
- En fonction de votre budget

Modèles d'équipe

- Two early philosophies
  - Decentralized/democratic
  - Centralized/autocratic
- Variation
  - Controlled Decentralized

Modèles d'équipe

- Business Team
  - Most common model
  - Technical lead + team (rest team at equal status)
  - Hierarchical with one principal contact
  - Adaptable and general
  - Variation: Democratic Team
    - All decisions made by whole team
    - See Weinberg's "egoless programming" model

Modèles d'équipe

- Chief-Programmer Team
  - From IBM in 70's
    - See Brooks and Mythical Man-Month
  - a.k.a. 'surgical team'
  - Puts a superstar at the top
    - Others then specialize around him/her
      - » Backup Programmer
      - » Co-pilot or alter-ego
      - » Administrator
      - » Toolsmith
      - » "Language lawyer"
  - Issues
      - » Difficult to achieve
      - » Ego issues: superstar and/or team
  - Can be appropriate for creative projects or tactical execution



Modèles d'équipe

- Skunkworks Team
  - Put a bunch of talented, creative developers away from the mother ship
    - Off-site literally or figuratively
  - Pro: Creates high ownership & buy-in
  - Con: Little visibility into team progress
  - Applicable: exploratory projects needing creativity
    - Not on well-defined or narrow problem

Modèles d'équipe

- SWAT Team
  - Highly skilled team
  - Skills tightly match goal
  - Members often work together
  - Ex: security swat team, Oracle performance team

Modèles d'équipe

- Large teams
  - Communication increases multiplicatively
    - Square of the number of people
    - 50 programmers = 1200 possible paths
    - Communication must be formalized
  - Always use a hierarchy
  - Reduce units to optimal team sizes
    - Always less than 10

Taille d'équipe

- What is the optimal team size?
  - 4-6 developers
    - Tech lead + developers
  - Small projects inspire stronger identification
  - Increases cohesiveness
  - QA, ops, and design on top of this



Références

- CNRS, DSI (http://www.dsi.cnrs.fr/conduite-projet/Default.htm)
- Association Francophone de la gestion de projet (http://www.afitep.fr/Default.htm)
- Project Management Institute (PMI) (http://www.pmi.org/)
- Software Engineering Institute (SEI) (http://www.sei.cmu.edu/)
- IEEE Software Engineering Group (http://standards.ieee.org/software/)
- Guide to the Software Engineering Body of Knowledge (http://www.swebok.org/)
- Cost estimation tools
  - http://www.retisoft.com/SCEPFeatures.html
  - http://www.construx.com/estimate
- FAQ on Function Points
  - http://ourworld.compuserve.com/homepages/softcomp/fpfaq.htm
- Choosing a project management tool
  - http://www.4pm.com/articles/selpmsw.html
  - http://www.infogoal.com/pmc/pmcswr.htm
- Project management tools
  - http://www.startwright.com/project1.htm
  - http://www.kidasa.com
  - http://www.criticaltools.com/pertmain.htm
  - http://www.guysoftware.com/planbee.htm
  - http://www.minuteman-systems.com
  - http://www.microsoft.com/office/project/prodinfo/default.mspx
  - http://www.eclipse-plugins.info/eclipse/plugins.jsp?category=Project+management

